Computer Aided Design 1

Practice

Bachelor in Computer Vision

Cédric Lemaitre
c.lemaitre58@gmail.com

Intro to Matlab

NOTE
For each problem you shall create a script, for example `problem1.m`, containing all commands to answer the questions.

Problem 1

Make the following variables

1. $a = [3.14\ 15\ 9\ 26]$
2. $b = \begin{bmatrix} 2.71 \\ 7 \\ 2.1 \\ 71 \end{bmatrix}$
3. $c = \begin{bmatrix} 5 \\ 4.8 \\ \vdots \\ -4.8 \\ -5 \end{bmatrix}$ (all the numbers from 5 to -5 in increments of -0.2).
4. $A = \begin{bmatrix} 2 & \dots & 2 \\ \vdots & \ddots & \vdots \\ 2 & \dots & 2 \end{bmatrix}$ a 9×9 matrix full of 2's (use the commands **ones** or **zeros**)
5. $B = \begin{bmatrix} 1 & 0 & \dots & & 0 \\ 0 & \ddots & 0 \ddots & & \\ \vdots & 0 & 5 & 0 & \vdots \\ & \ddots & 0 & \ddots & 0 \\ 0 & & \dots & 0 & 1 \end{bmatrix}$ a 9×9 matrix of all zeros, but with the values $[1\ 2\ 3\ 4\ 5\ 4\ 3\ 2\ 1]$ on the main diagonal, use **zeros** and **diag**.

6. $C = \begin{bmatrix} 1 & 11 & \dots & 91 \\ 2 & 12 & \dots & 92 \\ \vdots & \vdots & \ddots & \vdots \\ 10 & 20 & \dots & 100 \end{bmatrix}$ a 10×10 matrix where the vector 1:100 runs down the columns (use **reshape**)

7. Create a 5×5 matrix D of random integers with values on the range -3 to 3. Use **rand** and **floor** or **ceil**.

Problem 2

Solve the following equations using the variables created in **Problem 1**.

1. $x = \frac{1}{\sqrt{2\pi 2.5^2}} e^{-a^2/(2*2.5^2)}$
2. $y = \sqrt{(a^T)^2 + b^2}$
3. $z = \log_{10}(1/c)$, remember that $\log_{10}$ is the log base 10. So you use **log10** function.

Note that each of these variables is a vector of the right dimension.

Problem 3

If a matrix A is defined using the MATLAB code $A = [1\ 3\ 2;\ \ 2\ 1\ 1;\ \ 3\ 2\ 3]$, which command will produce the following matrix

$$B = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix}$$

Problem 4

Create the variables representing the following matrices:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 2 \\ -1 & 2 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 1 & 2 \end{bmatrix}$$

- Try performing the following operations: $A + B$, $A * B$, $A + C$, $B - A$, $A * C$, $C - B$, $C * A$. What are the results? What error messages are generated? Why?
- What is the difference between $A * B$ and $A. * B$?

Problem 5

All points with coordinates $x = r\cos(\theta)$ and $y = \cos(\theta)$, where r is a constant, lie on a circle with radius r. That is they satisfy the equation $x^2 + y^2 = r^2$.
Create a column vector for θ with the values, 0, $\pi/4$, $\pi/2$, $3\pi/4$, and $5\pi/4$. Take $r = 2$ and compute the column vectors x and y.
Now check that x and y indeed satisfy the equation of a circle, by computing the radius $r = \sqrt{(x^2 + y^2)^2}$.

Problem 6

The sum of geometric series $1 + r + r^2 + r^3 + \cdots + r^n$, approaches the limit $\frac{1}{1-r}$ for $r < 1$ as $n \to \infty$. Take $r = 0.5$ and compute the sums of series 0 to 10, 0 to 50, and 0 to 100. Calculate the aforementioned limit and compare with your summations. Use the built-in **sum** function.

Problem 7

The number of ways to choose k objects form a set of n objects is defined and calculated with the formula

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Define a Pascal matrix with the formula

$$P(i,j) = \binom{i+j-2}{i-1},$$

where i ranges from 1 to the number of rows and j ranges from 1 to the number of columns.
Use this definition and hand calculations to find a Pascal matrix of dimension 4×4.
Use Matlab's **pascal** command to check your result.

Problem 8

Read the documention of MATLAb's **primes** command, and use it to store the first 100 primes less than or equal to 1000.

- Fin the sum of the first primes
- Find the sum of the first, 20th and 97th primes.

Problem 9

Find a MATLAB one-line expression to cretae the $n \times n$ matrix A satisfying

$$a_{ij} = \begin{cases} 1 & \text{if } i - j \text{ is prime} \\ 0 & \text{otherwise} \end{cases}$$

Problem 10

Manipulating variables
For this problem, you need the file *classGrades.mat*.

- Open a script and name it *calculateGrades.m*. You'll write all the following command in this script.
- Load the *classGrades* file using the command **load**. The file contains a single variable called *namesAndGrades*.

- To see how *namesAndGrades* is structured, display the first 5 rows on your screen. The first column contains the students 'names', they are just integers from 1 to 15. The remaining 7 columns contain each student's score (on a sclae from 0 to 5) on each of 7 assignments. There are also some NaNs which indicates that a particular student was absent on that day and didn't do the assignment.
- We only care about the grades, so extract the submatrix containing all the rows but only columns 2 to 8 and name this new matrix *grades*. To make this work for any size matrix, don't hard-code the 8, but rather use **end** or **size** commands.

1. Calculate the mean score on each assignment. The result should be a 1x7 vector containing the mean grade on each assignment.

   First, do this using **mean** and display the mean grades you get. What's wrong with this result?

   Then use the **nanmean** command. What's different?

   Name this mean vector *meanGrades* (here you shoul use the vector without NaNs entries).

2. Now normalize each assignment so that the mean grade is 3.5. You'll want to divide each column of *grades* by the correct element of *meanGrades*.

   - Make a matrix called *meanMatrix* such that it is the same size as *grades*, and each row has the values *meanGrades*.
   - Calculate the curved grades as $curvedGrades = 3.5(grades/meanMatrix)$. Keep in mind that you want to do the division elementwise.
   - Compute and display the mean of *curvedGrades* to verify that they are all 3.5.
   - Because we divided by the mean and multiply by 3.5, it's possible that some grades that were initially close to 5 are now larger than 5. To fix this, find all the elements in *curvedGrades* that are greater than 5 and set them to 5. Use **find** command.

3. Calculate the total grade of each student and assign letter grades

   - To calculate the *totalGrade* vector, which will contain the numerical grade for each student, you want to take the mean of *curvedGrades* across the columns (use **nanmean**, see help for how to specify the dimension). Also, we only want to end up with numbers from 1 to 5, so calculate the ceiling of the *totalGrade* vector (use **ceil**).
   - Make a string called *letters* that contains the letter grades in increasing order: FDCBA
   - Make the final letter grades vector *letterGrades* by using *totalGrade* (which should only contain values between 1 and 5) to index into *letters*.
   - Finally, display the students grade using **disp**. You should find BCBBBACCBCCCCAB.